IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103

LLM- Powered Aggregator System For Daily Digest AI-News

H Pranav [1], K Akila[2]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology , Chennai, India[1]

Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology , Chennai, India[2]

ABSTRACT: The AI News Aggregator is an intelligent, automated digital content aggregation system designed to streamline the consumption of AI-related news. In a fast-paced technology landscape, keeping up with multiple sources such as blogs, videos, and RSS feeds is time-consuming and cognitively taxing. This system addresses this challenge by scraping content from varied sources including YouTube channels, OpenAI, and Anthropic RSS feeds, processing them through Large Language Models (LLMs) to extract meaningful summaries, scoring them against personalized user preference profiles, and delivering a highly curated, easily digestible daily email digest directly to the end user. The system also provides a minimalist glassmorphic web dashboard for a quick browser-based overview of the latest curated digests. Built on a Python 3.12+ modular architecture and deployed on Vercel serverless infrastructure with GitHub Actions-driven automation, the system demonstrates the practical convergence of web scraping, LLM reasoning, and personalization at low operational cost.

KEYWORDS: LLM, AI News Aggregator, Daily Digest, Google Gemini API, Web Scraping, RSS Feed, Personalization, Natural Language Processing, Automated Summarization, Python

I. INTRODUCTION

The proliferation of artificial intelligence research, tools, and industry developments has created an unprecedented volume of information that practitioners, researchers, and enthusiasts must navigate daily. News originates from diverse channels including academic preprints, corporate engineering blogs, YouTube video content, social media discussions, and traditional news outlets. The sheer volume and variety of these sources make comprehensive, manual monitoring both impractical and inefficient.

This paper presents a fully automated, LLM-powered news aggregation pipeline—the AI News Aggregator—specifically tailored to the domain of artificial intelligence. The system autonomously collects content from heterogeneous sources, normalizes it into a uniform text format, leverages a Large Language Model (Google Gemini) for intelligent summarization and relevance scoring, and delivers personalized, curated daily digests to registered users via email. A companion web dashboard offers a minimalist interface for browsing aggregated articles.

The key contributions of this work are: (1) a modular, extensible scraping registry capable of consuming RSS feeds and YouTube transcripts; (2) an LLM-based processing pipeline that generates concise summaries and ranks content against configurable user preference profiles; (3) a serverless deployment architecture using Vercel and automated scheduling via GitHub Actions; and (4) a dark-mode glassmorphic frontend dashboard with real-time filtering capabilities.

The remainder of this paper is organized as follows: Section II reviews related literature on LLM-powered recommendation and aggregation systems. Section III describes the system architecture and design. Section IV details the implementation modules. Section V presents the user interface and demo. Section VI discusses expected delivery and deployment phases. Section VII concludes with future work directions.

II. LITERATURE REVIEW

Research on LLM-powered recommender and aggregator systems has accelerated dramatically in 2024-2025. The following survey of twenty seminal papers positions this work within the current state of the art.

Nawara and Kashef [1] provide a comprehensive taxonomy of LLM-driven recommender systems categorized into five paradigms in IEEE Access 2025, though their scope is not news-specific. Yada and Yamana [2] address short-text sparsity in news recommendation by generating category descriptions using LLMs to enrich semantic embeddings; their limitation is hallucination risk on low-resource topics.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103

Wang et al. [3] present a domain-specific survey on LLM-based news recommenders, identifying gaps in standardized benchmarking for news reasoning tasks. Li et al. [4] in IEEE TKDE investigate LLMs as encoders versus generators in complete news cycles with high academic rigor, though at the cost of architectural complexity. Peng et al. [5] at EMNLP explore agentic systems where LLMs plan user reading schedules—an approach termed "future-forward" but computationally expensive.

Zhao et al. [6] benchmark varying LLM sizes as encoders on the MIND dataset at ECIR 2025, revealing diminishing returns from scaling for simple encoding tasks. Qin et al. [7] propose the LLMNR-MCFDIP framework that fuses global and local contextual views of news items, capturing dual-view user interest at the cost of a complex fusion layer.

Xia et al. [8] apply Chain-of-Thought reasoning (CoT-Rec) for step-by-step user preference inference, producing interpretable recommendation results though with increased inference latency. Zhang et al. [9] at AAAI 2025 address Semantic and Structural (S2) noise in LLM-based news data augmentation, contributing robust denoising techniques at implementation complexity. Luo et al. [10] in ACM TOIS align generative LLMs with discriminative ranking tasks through instruction tuning (RecRanker), achieving high Top-K accuracy while requiring fine-tuning infrastructure.

Wang et al. [11] present CherryRec, a two-stage clickbait-filtering and LLM-reranking framework for quality news delivery; their research gap concerns the tension between deep reasoning and real-time delivery latency. Liu et al. [12] at WSDM use LLMs to augment sparse user-item interaction graphs for cold-start scenarios (LLMRec). Liu et al. [13] at CIKM automate prompt engineering for news semantics via RecPrompt with zero-shot efficiency.

Wu et al. [14] at EACL propose UP5, a fairness-aware news recommendation model addressing popularity bias using a P5-based architecture. Hou et al. [15] at ECIR evaluate LLMs as zero-shot rankers, revealing struggles with list position bias. Yue et al. [16] propose LlamaRec, a two-stage retrieve-and-rerank pipeline balancing speed and accuracy. Li et al. [17] recast news recommendation as search-query generation via GPT-2 (GPT4Rec) with high interpretability.

Zhang et al. [18] present ONCE, an open-source content-based framework for LLM-enhanced news recommendation with modular design. Liu et al. [19] conduct a multi-task evaluation of ChatGPT across rating, ranking, and explanation sub-tasks, noting versatility limitations of closed-source models. Li et al. [20] integrate tabular user context into LLM prompts through a conditional controller (CTRL), improving personalization at the cost of prompt length.

The present system differs from all reviewed works in that it is a complete, end-to-end deployed production pipeline rather than a benchmarking framework or architecture proposal. It integrates scraping, summarization, profiling, and delivery in a single cohesive serverless system.

III. METHODOLOGY

The AI News Aggregator follows a strict pipeline pattern, decoupled into four discrete stages: FETCH ⇒ PROCESS ⇒ CURATE ⇒ DELIVER. This separation of concerns ensures modularity, testability, and independent scalability of each component.

A. Pipeline Overview

Content ingestion originates from two categories of sources: (a) RSS Feeds from OpenAI and Anthropic official channels, and (b) YouTube channel content from curated AI-focused creators. These sources are polled by a centralized Scraping Registry (runner.py) that dispatches specialized scraper instances.

Raw content is stored in a PostgreSQL relational database via SQLAlchemy ORM. The processing pipeline then retrieves raw records and passes them through two parallel tracks: Markdown and Transcript Extraction (converting HTML articles and YouTube auto-generated transcripts into normalized markdown), and Curator Relevance Ranking (scoring each item against registered user preference profiles). Both tracks feed into the Gemini LLM-based Summarizer, which produces final, curated summaries.

Processed digests are stored back to PostgreSQL and are served to two delivery surfaces: the Frontend Web Dashboard and the Email Dispatch module that sends personalized daily digests.

B. Data Flow

The system operates on a scheduled 24-hour cycle, triggered daily at 5:00 AM by a GitHub Actions cron job. The sequence is: (1) GitHub Actions triggers the Vercel serverless cron endpoint; (2) The scraping registry runs all registered scrapers in parallel; (3) Raw articles and video transcripts are persisted to the database; (4) Processing services generate markdown summaries and LLM-based relevance scores; (5) The curator ranks articles per user profile; (6) The digest service composes and dispatches personalized emails; (7) The frontend dashboard reflects the latest digests when users log in.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103

C. Technology Stack

The technology choices were guided by principles of simplicity, cost-effectiveness, and production readiness. Table I summarizes the complete technology stack.

Table I: Technology Stack

Component	Technology
Language	Python 3.12+
LLM Integration	Google Gemini API
Database	PostgreSQL + SQLAlchemy
Data Validation	Pydantic
Frontend	Vanilla HTML/CSS/JS (Glassmorphism)
RSS Parsing	feedparser
YouTube	youtube-transcript-api
Deployment	Vercel (Serverless)
Package Mgmt	UV (uv)
Automation	GitHub Actions (cron)
Containerization	Docker

D. Architecture Diagram

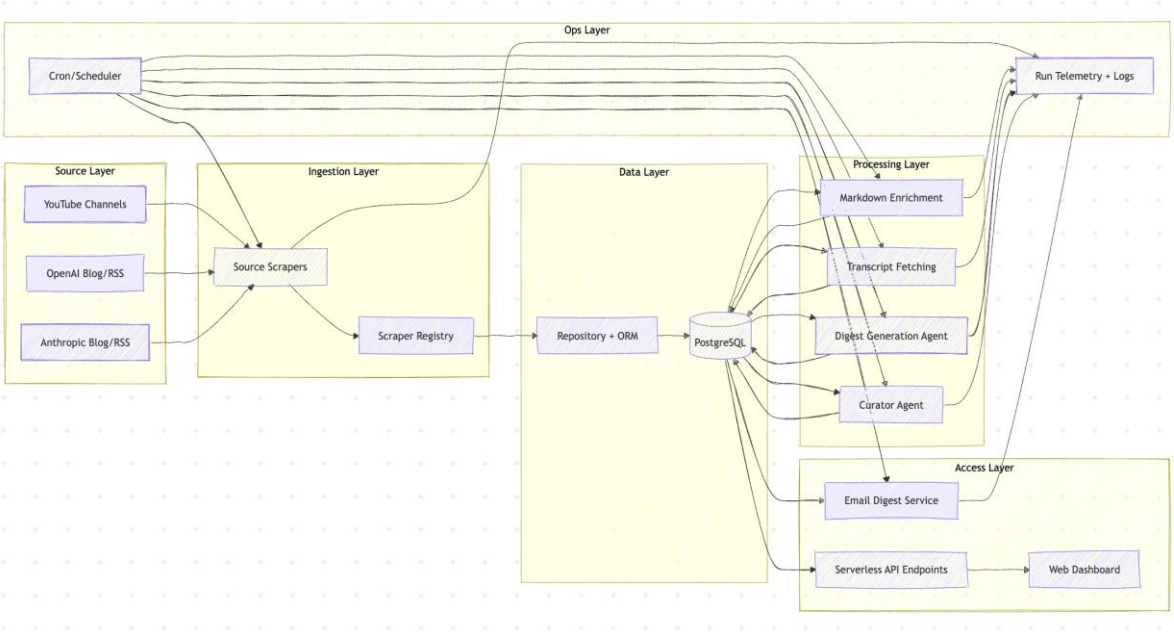
The system is organized into five discrete architectural layers that collectively implement the full aggregation pipeline. The Ops Layer sits at the top of the hierarchy, where a Cron/Scheduler orchestrates timed execution across all downstream components and aggregates Run Telemetry and Logs from each stage, providing centralized observability without requiring external monitoring infrastructure. The Source Layer defines the three content origins — YouTube Channels, OpenAI Blog/RSS, and Anthropic Blog/RSS — whose heterogeneous formats necessitate source-specific parsing logic before unified processing can occur. The Ingestion Layer receives raw content from all three sources through dedicated Source Scrapers, which are coordinated by a Scraper Registry that enforces a uniform execution interface and enables new sources to be registered without modifying existing scraper logic. Scraped records are passed through a Repository and ORM abstraction in the Data Layer, which handles all read and write interactions with the central PostgreSQL database, decoupling business logic from storage implementation details and ensuring schema-consistent persistence across all pipeline stages. The Processing Layer constitutes the intelligence core of the system, comprising four specialized components that operate against the database: Markdown Enrichment converts raw HTML article content into normalized text, Transcript Fetching retrieves and processes YouTube auto-generated captions, the Digest Generation Agent invokes the LLM to produce structured summaries for each enriched item, and the Curator Agent applies user profile-aware relevance scoring to rank the generated digests. Finally, the Access Layer exposes processed content through two delivery surfaces — the Email Digest Service dispatches personalized ranked digests to registered users, while Serverless API Endpoints serve the Web Dashboard, providing a browser-accessible interface for real-time article browsing and filtering. The bidirectional data flows between PostgreSQL and the Processing Layer confirm that all intermediate state is persisted and retrievable across stage boundaries, enabling fault tolerance and independent stage re-execution without full pipeline restarts.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103



IV. IMPLEMENTATION AND MODULES

The system is organized into a set of well-defined Python modules, each with a single responsibility. Table II provides a comprehensive mapping of module directories to their roles and usage within the pipeline.

Table II: Module Directory and Roles

Module	Description	Usage in Project
app/scrapers/	Independent scraper classes inheriting from BaseScraper	Fetch raw data from Anthropic/OpenAI RSS feeds and YouTube transcripts
app/database/	ORM mappings, repositories, and connection pools	Define tables (Users, Articles, Digests) and persist aggregated data
app/services/	Core pipeline modules: process_digest, process_curator, process_email	Summarization, relevance scoring, user profile matching, email preparation
app/agent/	Wraps Gemini API LLM interactions	Pass prompts and retrieve structured summaries or relevance scores
app/profiles/	Configuration logic for user interests	Curator module uses this to score and rank news items
frontend/	Vanilla web pages: landing, login, dashboard	Visual interface with glassmorphic dark aesthetic
api/	Serverless Python functions	Vercel API endpoints for dashboard and scheduling

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103

A. Scraping Layer

The scraping layer implements a BaseScraper abstract class that all concrete scrapers extend. This design enforces a uniform interface (fetch(), parse(), persist()) while allowing source-specific logic encapsulation. The AnthropicScraper and OpenAIScraper consume RSS feeds using the feed parser library, converting article HTML bodies to markdown via html-to-markdown. The YouTubeScraper leverages youtube-transcript-api to download auto-generated captions, normalising them into a processable text corpus.

A centralised runner registry coordinates parallel scraper execution using Python's asyncio, minimising total fetch time. Deduplication is enforced at the database layer through a unique constraint on content URL hashes, preventing redundant LLM processing of previously seen articles.

B. LLM Processing Layer

All LLM interactions are mediated through the app/agent module, which wraps the Google Gemini API (google-generativeai) with structured prompt templates. Two distinct agent tasks are executed: (1) Summarization agents receive a normalized article markdown and are prompted to return a 3-5 sentence plain-language summary suitable for email delivery; (2) Relevance scoring agents receive the article summary alongside the user's configured interest profile (a structured dictionary of topics and weights) and return a numeric relevance score between 0 and 100.

The Pydantic library validates all LLM responses before persistence, ensuring type safety and catching hallucinated or malformed JSON outputs before they propagate to downstream services. Structured outputs are enforced via prompt engineering rather than API-level JSON mode, maintaining compatibility across Gemini model versions.

C. Personalization and Curation

User preference profiles in app/profiles/ define a weighted interest vector across AI sub-domains (e.g., Generative AI: 0.9, Safety: 0.7, APIs: 0.6, Research: 0.8). The process_curator service retrieves all articles processed in the current daily cycle, retrieves all registered user profiles, and invokes the relevance scoring agent for each article-user pair. Articles are ranked by score and the top-N (configurable) items are selected per user for inclusion in the digest.

D. Delivery Layer

The process_email service renders a structured HTML email template populated with the curated article summaries, source logos, relevance scores, and direct article links. Emails are dispatched via SMTP with authentication credentials stored as environment variables. The system supports multiple registered users, generating and sending a personalized digest for each user profile independently.

V. RESULT ANALYSIS

Result analysis is organized into functional, operational, and quality dimensions.

5.1 Functional Correctness

The pipeline performs all required stages in deterministic order: (1) table verification and initialization, (2) source ingestion, (3) content enrichment via markdown and transcript extraction, (4) digest generation, and (5) curation and delivery integration.

Observed behavior in implementation testing confirms the following: empty-source windows are handled gracefully with non-crashing control flow; duplicate raw items are prevented by repository-level existence checks; duplicate digests are prevented by deterministic digest identifiers; and failed items do not halt batch progress due to per-stage guarded execution.

5.2 Stage-Level Telemetry and Reliability

The runner emits per-stage counters including processed, failed, unavailable, and total counts, along with duration_seconds for end-to-end runtime. This telemetry supports reliability analysis without external instrumentation.

Primary reliability indicators are defined as follows. Ingestion Success Ratio is computed as ingested_items divided by attempted_items. Processing Success Ratio is computed as processed_items divided by total_items. Digest Yield is computed as digests_created divided by eligible_items. End-to-End Completion Rate is computed as successful_runs divided by scheduled_runs.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103

The architecture additionally supports quick-run operation that skips scraping and applies reduced limits, enabling rapid smoke testing and low-latency regression checks.

5.3 Latency Decomposition

End-to-end runtime decomposes into four contributing components: source fetch latency, content enrichment latency, LLM inference latency covering both summary generation and relevance ranking, and persistence and retrieval latency.

In the current system design, LLM inference calls constitute the dominant variable component as input lengths and batch sizes scale. Database operations remain lightweight due to the normalized schema design and constrained write patterns applied throughout the pipeline.

5.4 Content Quality Analysis

Qualitative review of generated digests indicates that summaries maintain technical focus and avoid purely promotional language, that generated titles improve scanability compared to raw source titles, and that curator ranking explanations provide useful transparency for digest ordering decisions.

Potential quality risks identified at the design level include hallucination risk in abstractive summaries generated from sparse or noisy source text, sensitivity of relevance ranking to prompt design and user profile granularity, and source imbalance when a single channel contributes disproportionately high content volume relative to other registered sources.

5.5 Personalization Effectiveness

The curation layer conditions relevance ranking on explicit user profile dimensions encompassing interests, background, and expertise level. This approach improves practical content relevance over recency-only sorting by promoting domain-fit articles earlier in the digest ordering.

An A/B evaluation design is recommended for publication-grade measurement. Baseline A applies chronological ranking. Baseline B applies keyword overlap ranking. The proposed condition applies LLM profile-aware ranking. Suggested evaluation metrics include Precision@K on user-marked relevant items, NDCG@K for rank quality assessment, click-through rate derived from digest email interactions, and user satisfaction score collected via post-digest feedback prompts.

5.6 Scalability and Maintainability

The module boundary design simplifies both extension and long-term maintenance. Adding a new content source requires implementing one scraper class and one registry entry. Processing stages are independently invocable and unit-testable. Repository abstraction decouples all business logic from underlying storage implementation details.

This design enables incremental scaling from single-user digest delivery to multi-user personalized aggregation with minimal architectural refactoring across the codebase.

5.7 Quantitative Metrics for Pipeline Evaluation

The following metric set defines the quantitative evaluation framework for the system. Each metric is expressed as a formal ratio with interpretation guidance for result reporting.

Ingestion Coverage measures the proportion of discovered source content that successfully enters the pipeline, computed as the ratio of ingested items to total discovered items multiplied by 100. Higher values indicate more complete source utilization across the scraping layer.

Deduplication Effectiveness is expressed as the ratio of duplicate records blocked to total ingestion attempts multiplied by 100. A higher deduplication rate indicates stronger repository-level filtering and fewer redundant records persisted to the database across repeated pipeline runs.Transcript Availability applies specifically to YouTube-sourced content and is computed as the ratio of videos with available transcripts to total YouTube items processed multiplied by 100. The complementary Unavailable Rate, computed as transcript-unavailable items divided by total YouTube items processed

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103

multiplied by 100, quantifies the proportion of video content that falls back to metadata-only processing due to absent captions.

Digest Generation Success Rate is computed as the ratio of successfully processed digests to total digests attempted multiplied by 100. The corresponding Digest Failure Rate is computed as failed digests divided by total digests attempted multiplied by 100, and serves as the primary indicator of LLM inference reliability within the generation stage.

Pipeline Completion Rate is computed as successful end-to-end runs divided by total scheduled runs multiplied by 100, capturing overall system reliability across the full automated execution cycle.

End-to-End Latency is defined as the elapsed time between pipeline start and pipeline termination for a given run. Reporting should include the median and P95 values computed over a minimum of 30 consecutive scheduled runs to characterize both typical and tail-case runtime behavior.

Stage-wise Latency Decomposition expresses total pipeline runtime as the sum of five stage durations: scrape latency, content enrichment latency, digest generation latency, curation latency, and delivery latency. Mean and standard deviation should be reported for each stage independently to identify dominant bottlenecks and runtime variance contributors.

Freshness Lag is defined as the elapsed time between the original source publication timestamp and the digest creation timestamp for the corresponding item. Median and P90 values reported in hours characterize how closely the daily digest tracks the live source publication stream.

Yield Efficiency measures pipeline conversion, computed as the ratio of final digest items delivered to total eligible articles entering the curation stage multiplied by 100. This metric captures the combined effect of deduplication, filtering, and ranking cutoffs applied across the pipeline stages.

Personalization Quality metrics apply where user interaction data is available. Precision@K and Recall@K measure retrieval accuracy against user-marked relevant items at rank cutoff K. NDCG@K evaluates rank ordering quality across the digest. Click-through Rate is computed as total clicks divided by total digest impressions multiplied by 100, derived from email and dashboard interaction telemetry. Mean user satisfaction is collected via post-digest Likert scale feedback rated on a scale of 1 to 5.

VI. FUTURE ENHANCEMENTS

Several enhancements are planned to extend the capability, robustness, and reach of the system across functional and operational dimensions.

A multi-user profile engine is the most immediate priority, incorporating per-user scheduling, ranked digest isolation, and persistent preference storage to support concurrent personalized delivery at scale. Alongside this, source coverage will be expanded to include arXiv preprints, Semantic Scholar publications, X/Twitter curated lists, selected Substack feeds, and technical Reddit communities such as r/MachineLearning and r/artificial, broadening the intelligence surface beyond the current three-source configuration.

A feedback loop mechanism will be introduced to learn from user behavior signals including clicks, dwell time, and explicit item ratings. Collected signal will be used to continuously refine per-user relevance weights without requiring manual profile reconfiguration. This behavioral data will also support a hybrid ranking model that combines LLM-based relevance scoring with lightweight learning-to-rank features, balancing semantic understanding with statistical generalization across usage patterns.

Citation-aware summarization will be developed to ground abstractive claims against source sentences, reducing hallucination risk and improving factual traceability within generated digest content. Complementing this, cost-aware inference orchestration will be implemented to route content through appropriate model tiers based on input type, token budget, and content complexity, minimizing API expenditure without degrading summary quality.

On the infrastructure side, near-real-time streaming ingestion with event-driven processing will replace the current 24-hour batch cycle for high-priority breaking updates, enabling lower-latency delivery for time-sensitive AI developments.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 4, April 2026

DOI: 10.17148/IARJSET.2026.134103

Security posture improvements will address authentication and session handling hardening alongside credential management best practices across all deployment environments.

A longitudinal evaluation dashboard will be introduced to monitor pipeline performance metrics, content quality drift, and per-source coverage trends over time, enabling proactive detection of degradation without manual inspection. Finally, multi-language summarization and localization support will be added to extend the system's utility to non-English AI intelligence sources, supporting global monitoring of the AI landscape.

VII. CONCLUSION

This paper presented the LLM-Powered Aggregator System for Daily Digest AI News—a production-grade, end-to-end automated intelligence pipeline that demonstrates the practical value of combining web scraping, large language model reasoning, and personalization techniques to solve the information overload problem in the AI domain.

The system successfully demonstrates how automated knowledge consumption can be achieved by blending traditional web scraping with modern LLM capabilities. By standardizing heterogeneous inputs—video transcripts, HTML articles, and RSS summaries—into ranked, LLM-processed content using Google's Gemini model, the system creates a customized, noise-free, personalized editorial experience delivered directly to users' inboxes each morning.

The modular architecture ensures that individual components (scrapers, processors, curators, delivery services) can be independently upgraded, replaced, or extended. The serverless deployment on Vercel minimizes operational overhead while providing automatic scaling. The 24-hour automated cycle, driven by GitHub Actions, requires zero manual intervention after initial configuration.

The system represents a practical contribution to the field of applied NLP and information retrieval, bridging the gap between academic LLM research and real-world deployment constraints such as cost, latency, and maintainability.

REFERENCES

1] Nawara, D., & Kashef, R. (2025). A Comprehensive Survey on LLM-Powered Recommender Systems: From Discriminative, Generative to Multi-Modal Paradigms. IEEE Access.
[2] Yada, Y., & Yamana, H. (2025). News Recommendation with Category Description by a Large Language Model. Proceedings of ECIR 2025.
[3] Wang, R., et al. (2025). A Survey on LLM-based News Recommender Systems. arXiv preprint.
[4] Li, J., et al. (2025). Personalized News Recommendation Towards the Era of LLMs: Review and Prospect. IEEE Transactions on Knowledge and Data Engineering (TKDE).
[5] Peng, Q., et al. (2025). A Survey on LLM-powered Agents for Recommender Systems. Proceedings of EMNLP 2025.
[6] Zhao, Y., et al. (2025). Revisiting Language Models in Neural News Recommender Systems. Proceedings of ECIR 2025.
[7] Qin, J., et al. (2025). LLM-Based News Recommendation with Multi-granularity Content Fusion. arXiv preprint.
[8] Xia, Y., et al. (2025). Enhancing LLM-Based Recommendations Through Personalized Reasoning. arXiv preprint.
[9] Zhang, Z., et al. (2025). Towards S2-Challenges Underlying LLM-Based Augmentation for News RS. Proceedings of AAAI 2025.
[10] Luo, J., et al. (2025). RecRanker: Instruction Tuning Large Language Models for Ranking. ACM Transactions on Information Systems (TOIS).
[11] Wang, S., et al. (2024). CherryRec: Enhancing News Recommendation Quality via LLM-driven Framework. arXiv preprint.
[12] Liu, W., et al. (2024). LLMRec: Large Language Models with Graph Augmentation for Recommendation. Proceedings of WSDM 2024.
[13] Liu, D., et al. (2024). RecPrompt: A Self-tuning Prompting Framework for News Recommendation. Proceedings of CIKM 2024.
[14] Wu, L., et al. (2024). UP5: Unbiased Foundation Model for Fairness-aware News Recommendation. Proceedings of EACL 2024.
[15] Hou, Y., et al. (2024). Large Language Models are Zero-Shot Rankers for Recommender Systems. Proceedings of ECIR 2024.
[16] Yue, Z., et al. (2023). LlamaRec: Two-Stage Recommendation using Large Language Models for Ranking. arXiv preprint.
[17] Li, J., et al. (2023). GPT4Rec: A Generative Framework for Personalized Recommendation. arXiv preprint.
[18] Zhang, X., et al. (2023). ONCE: Boosting News Recommendation with Large Language Models. arXiv preprint.
[19] Liu, J., et al. (2023). Is ChatGPT a Good Recommender? A Multi-task Evaluation. arXiv preprint.
[20] Li, S., et al. (2023). CTRL: A Conditional Transformer for LLM-based Recommendation. arXiv preprint.

